

Practising Agile Development for Beginners

04.11.2007

Lars Jankowsky

CTO OXID eSales AG

About me:

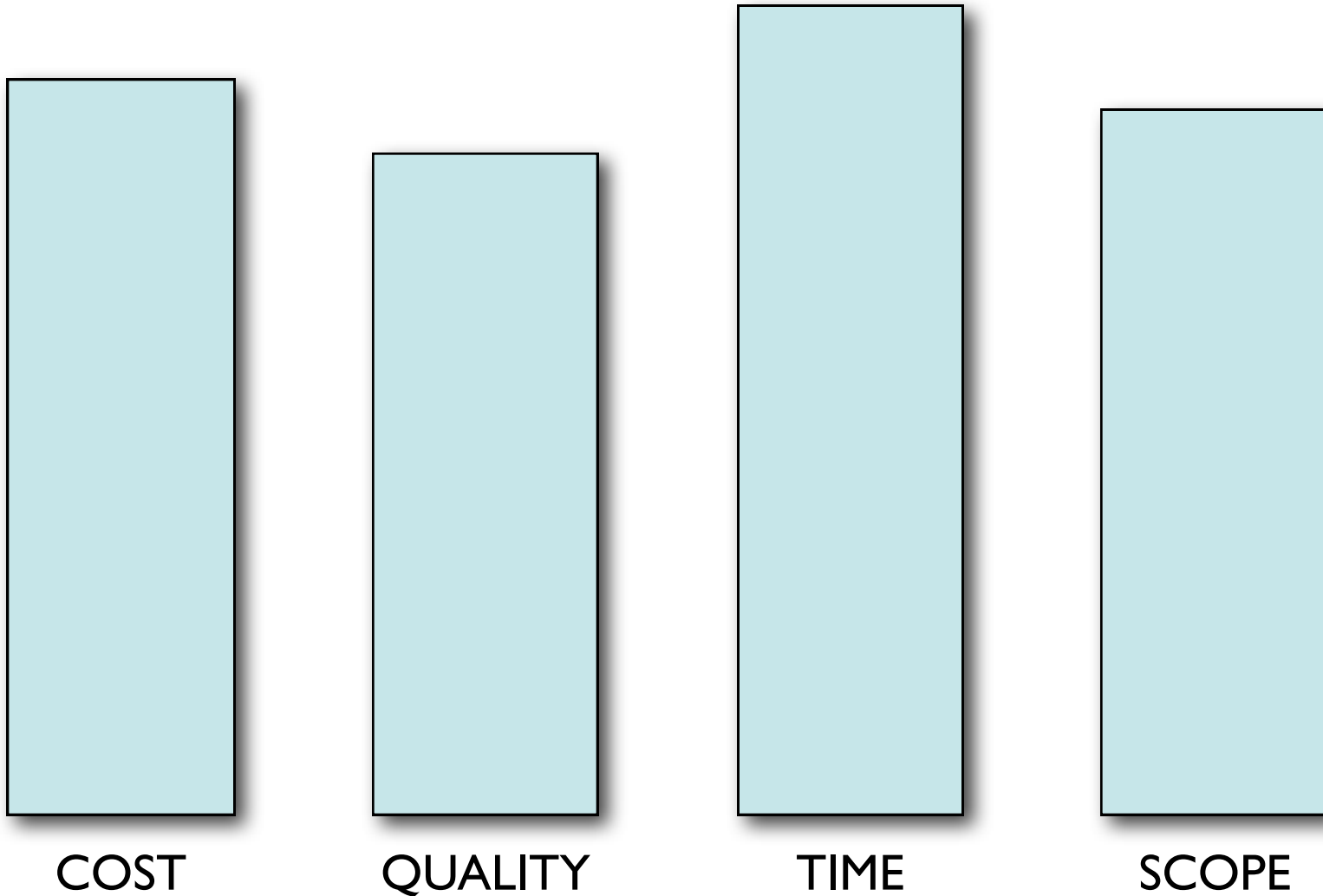
PHP, C++, Developer, Teamleader since 1992

PHP since 1998

Many successful projects from 2 to 20 developers

Running right now three projects using eXtreme Programming

lessons learned



Get Ready

FIRE!

Aim...

Aim...

Aim...

Agile methods are based on:

+

Communication

+

Simplicity

+

Feedback

+

Courage

<http://agilemanifesto.org/>

**Individuals and interactions over processes
and tools**

**Working software over comprehensive
documentation**

Customer collaboration over contract negotiation

Responding to change over following a plan

Agile ?

THINK !

popular Agile methods

- Adaptive Software Development
- Feature Driven Development
- DSDM - Dynamic Systems Development Method
- Scrum
- crystal clear
- XP
-

Adaptive Software Development

- Speculate
- Collaborate
- Learn
- <http://www.adaptivesd.com/>

Feature Driven Development

- Develop Overall Model
- Build Feature List
- Plan By Feature
- Design By Feature
- Build By Feature
- http://en.wikipedia.org/wiki/Feature_Driven_Development

DSDM - Dynamic Systems Development Method

- based on Rapid Application Development (RAD)
- User involvement
- The project team must be empowered
- Development is iterative and incremental
- Testing is carried out throughout the project life-cycle
- Communication and cooperation
- http://en.wikipedia.org/wiki/Dynamic_Systems_Development_Method

Scrum

- Plan (product backlog)
- Sprint planning session
- Sprint (== Iteration)
- Daily meetings
- Sprint review
- Closure
- <http://www.controlchaos.com/>

crystal clear

- Frequent Delivery
- Reflective Improvement
- Osmotic Communication
- Personal Safety
- Focus
- Easy Access to Expert Users
- Technical Environment with Automated Tests, Configuration Management, and Frequent Integration
- <http://www.informit.com/articles/article.asp?p=345009&seqNum=1&rl=1>

eXtreme Programming

- for smaller teams (2 - 12)
- focuses on automatic testing
- a change in the way we program
- includes continuous integration
- <http://www.extremeprogramming.org/>

Agile methods ?

Adaptive Software Development	DSDM	Scrum	crystal clear	eXtreme Programm.
Speculate	User involvement	Plan	Frequent Delivery	Communicate
Collaborate	Team must be empowered	Planning session	Reflective Improvement	Small Teams
Learn	Development is iterative	Sprint (Iteration)	Communicate	Autom. Tests
	Testing full life-cycle	Daily meetings	Personal Safety	cont. Integration
	Communicate	Sprint review	Automated Tests	Courage

Agile methods are based on:

+

Communication

+

Simplicity

+

Feedback

+

Courage

lessons learned

let's play the game

eXtreme Programming

“Software development is too hard to spend time on things that don't matter. So, what really matters? Listening, Testing, Coding, and Designing.”

(Kent Beck, “father” of Extreme Programming)

eXtreme Programming



embrace changes!



people vs. hardware



automated tests



continous integration

eXtreme Programming

+

Planning

+

Designing

+

Coding

+

Testing

Planning XP



The Customer



Stories



Estimation



Release Plan

Planning XP - the Customer

- Is always available
- Writes User Stories and specifies Functional Tests
- Sets priorities, explains stories
- May or may not be an end-user
- Has authority to decide questions about the stories
- Create/Defines acceptance tests

Balancing Power

Customer bill of rights

As the customer, you have the right to:

An overall plan, to know what can be accomplished, when, and at what cost;
Get the most possible value out of every programming week;
See progress in a running system, proven to work by passing repeatable tests that you specify;
Change your mind, to substitute functionality, and to change priorities without paying exorbitant costs;
Be informed of schedule changes, in time to choose how to reduce scope to restore the original date, even cancel at any time and be left with a useful working system reflecting investment to date.

Programmer bill of rights

As the Developer, you have the right to:

Know what is needed, with clear declarations of priority;
Produce quality work at all times;
Ask for and receive help from peers, superiors, and customers;
Make and update your own estimates;
Accept your responsibilities instead of having them assigned to you.

Who is our customer ?

Planning XP - the Stories

- A short story about the functionality from the point of view of the Customer
- No technical jargon
- One for each major feature in the system
- Must be written by the customer
- Are used to create time estimates for release planning
- Replace a large Requirements Document
- The stories are the base for acceptance tests
- Only enough detail to make a low risk estimate of how long the story will take to implement.

Planning XP - the Stories

- Use simple technics (postit)
- Find a place where all developer can see the stories

User Story Example:

User should be able to register himself to get access to the system.

3 SP

Let's create our stories

Planning XP - Estimation

- Yesterdays Weather
- use your experience
- ask other teams which may have done similar things
- Estimate using „Story Points“ == ideal Person Days

Planning XP - Release Plan

- Customer defines the business value of desired stories (priority)
- Stories which are too large need to be split into smaller chunks
- Higher risks stories should come first
- Define release dates - these are fixed, scope not

The Release plan

4	Release Backlog for Release 1	SP
5	SEO: Google release (time buffer for small adjustments and fixes)	0,25
6	LCA: Error handling and mails, sms	0,5
7	LCA: Statistics	
8	Top 5 slowest/fastest batches	0,5
9	Number of batches sent/received per hour (factor 1)	1
10	Number of flights in the database	0,5
11	AGENT: getResult model - add batch status info	1
12	AGENT: verify model oneway flights	0,5
13	AGENT: verify model bothway flights	1
14	AGENT: LCA Http Interface	
15	verify oneway flights	0,5
16	getResult - add error info	1
17	getResult - add batch info	1
18	verify bothway flights	1

Never slip a date!

falling behind ?

Change the plan!

Let's make our release plan

Planning XP - Iterations

- From release backlog the stories with highest priorities are taken and assigned to the next iteration.
- An iteration can be 1-3 weeks usually
- Stories for Iteration are broken down into Tasks by Developers
- Tasks are estimated by all Developers as a group
- Developers “sign up” for Tasks, and estimate the time to complete

Planning XP - Iterations...

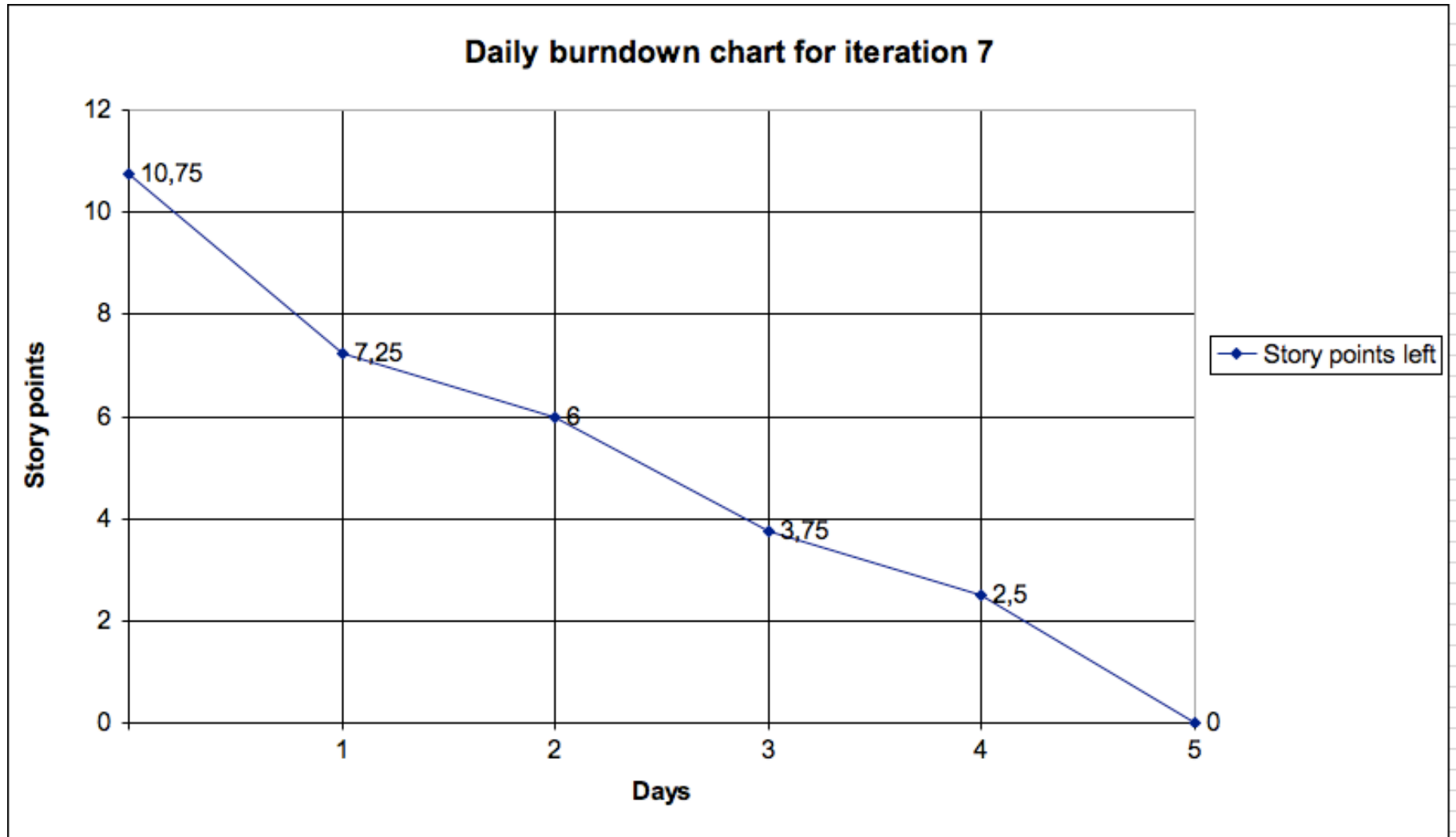
- If a task has more than 2-3 Story points, then break it into smaller parts.
- If there are any questions ask the customer, he should attend Iteration planning meeting
- Can only sign up for as many points as were completed in the last Iteration
- Once development begins, Project Velocity measures progress

Let`s plan our Iteration

During Iteration: Stand Up Meeting

- Track status in daily Stand-Up Meeting
- Ask for Story Points left - not for SP done
- Every morning for 5 minutes
- STAND UP!
- What did you do yesterday ?
- What do you plan todo today ?
- Any „Showstoppers“ or important issues ? (e.g. taking half day off...)
- Tell Team what you did, not some „Task number“

During Iteration: Tracking speed



Designing XP

+

Simplicity

+

Choose a System Metaphor

+

CRC Cards

+

Spike Solutions

+

Never Add Functionality Early

+

Refactor Mercilessly

Designing XP: Simplicity

- Keep things as simple as possible as long as possible by never adding functionality before it is scheduled.
- Always do the simplest thing that could possibly work
- Beware, keeping a design simple is hard work!

Designing XP: System Metaphor

- Name classes and methods consistently.
- Choose a system of names for your objects that everyone can understand easily
- Being able to guess at what something might be named if it already existed and being right is a real time saver

Designing XP: CRC Cards

- CRC == Class, Responsibilities, and Collaboration
- Use simple cards or Postit to define architecture
- A card can be a object, service etc.

Where are our CRC cards ?

Designing XP: Spike Solutions

- Don't guess on difficult questions, try it!
- Create simple programs to get answers.
- It reduces the risk of a technical problem or increase the reliability of a user story's estimate.

Any spike solution needed ?

Never Add Functionality Early

- Don't implement what you don't need now
- Only 10% of your guesses will be really used later
- Turn a blind eye towards future requirements and extra flexibility.
- Concentrate on what is scheduled for today only.

Designing XP: Refactor Mercilessly

“Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure.” (Martin Fowler)

Designing XP: Refactor Mercilessly

- No fear! Your tests will show if you break something.
- Keep your code clean!

System Metaphor?

CRC Cards?

Spike Solutions?

Coding XP

+

Use coding standards.

+

Integrate often. Check in daily.

+

Code the unit test first.

+

Use collective code ownership.

+

Leave optimization till last.

+

No overtime.

Coding XP: Use coding standards

- Agree on standards in team
- Coding standards keep the code consistent and easy for the entire team to read and refactor.
- Write them down somewhere (wiki / trac)

Coding XP: Check in daily

- Forces developer to write small functions
- Makes it impossible to create huge frameworks/functions
- This is the fundament for having good tests later

Coding XP: Code the unit test first.

- Upon creation of a function write unit tests
- Write many tests for each function
 - Success case
 - Error case
 - Stupid input case

Coding XP: Unit tests

- PHPUnit - <http://www.phpunit.de/>
- SimpleTest - <http://simpletest.sourceforge.net/>

Coding XP: collective code ownership

- Collective Code Ownership encourages everyone to contribute new ideas to all segments of the project. (Don Wells)
- Have a look what your colleagues are doing
- Any developer can change any line of code to add functionality, fix bugs, or refactor.

Coding XP: Pair Programming?

- All code to be included in a production release is created by two people working together at a single computer... ????
- Our experience is different!
- Use pair programming on difficult tasks e.g. refactoring, framework etc.
- In small teams it makes no sense to use it always.

Coding XP: Optimize last

- Make it work, make it right, then make it fast.
- Never guess what performance will be - measure it!
- Create clear rules for performance and test it. („ab“ or „siege“ can be used in unit tests)

Coding XP: No overtime

- Projects that require overtime to be finished on time will be late no matter what you do.
- Overtime sucks the spirit and motivation out of a team.
- Better play a game instead and build castles instead of creating bugs in the software

Testing XP

+

Create acceptance tests.

+

Use continuous integration!

+

Run tests automatically

+

Publish the results

Hope vs. knowledge

Testing XP: Acceptance Tests

- use Selenium
- Java solution which enables automatic „click“ tests.
- <http://www.openqa.org/selenium/>

Testing XP: Selenium

- Download Selenium RC
 - <http://www.openqa.org/selenium-rc/download.action>
- Create Acceptance Tests with Selenium IDE
- Use PHPUnit and create Unit Tests
- Base Tests on Customer Stories

Testing XP: continuous integration

Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. (Martin Fowler)

<http://www.martinfowler.com/articles/continuousIntegration.html>

Testing XP: continuous integration

- Various tools:
 - phing
 - XINC (<http://sourceforge.net/projects/xinc>)
- We use cruisecontrol
- check out <http://buildix.thoughtworks.com/>

Testing XP: cruisecontrol

- Download and install cruisecontrol
 - <http://cruisecontrol.sourceforge.net/gettingstartedbindist.html>
- Delete demo project files
- Create projects (/projects, /logs)
- Create svn/cvs user for cruisecontrol
- Manually checkout from svn into project dir
- Adapt config.xml, build.xml
- Check cruisecontrol.log for debugging

Testing XP: cruisecontrol GUI

cruisecontrol
continuous integration toolkit

Project
platforms

waiting for next time to build since
05/21/2007 09:26:08

Latest Build
05/21/2007 09:22:49 (build.348)
05/20/2007 10:51:22 (build.347)
01/26/2007 13:47:37 (build.346)
01/26/2007 13:37:04 (build.345)
01/26/2007 13:30:14 (build.344)
01/26/2007 13:23:36 (build.343)
01/26/2007 13:17:13 (build.342)
01/26/2007 13:12:47 (build.341)
01/26/2007 12:54:20 (build.340)
01/26/2007 12:49:48 (build.339)

More builds

RSS

Name	Status	Time(s)
.Unit_pfUserTest		
testLoad	Success	0.031
testAssign	Error »	
testDoubleAssign	Error »	
testHtmlSpecialChars	Error »	
testInsert	Error »	
testUpdate	Error »	
testLogin	Success	0.013
testSessionLogin	Failure »	
testDelete	Error »	
Properties »		
.Acceptance_UserPageTest		
testWrongEmailAddress	Success	10.945
testCreateUser	Failure »	
testLogout	Error »	
testRegisterWrongData	Success	14.704
testTTTStart	Error »	
Properties »		
.Unit_pfSOAPTest		
testLoginCorrectData	Success	0.252
testLoginWrongData	Success	0.085
testCreateUser	Failure »	
testCreateInvalidUser	Failure »	
testUpdateUserWithUserName	Failure »	
testUpdateUser	Failure »	
testKillSession	Failure »	
testGetMemberInfo	Success	0.329
testTakeTTT	Success	0.214
testQueryForMembers	Success	1.156

Questions ?

Resources used:

- Extreme Programming: A gentle introduction.
- <http://www.extremeprogramming.org/>

- Extreme Programming - Introduction
- Mayford Technologies Inc.

- Wikipedia

- Extreme Programming Explained: Embrace Change.
- Kent Beck

Contact information

Lars Jankowsky

CTO

OXID eSales AG

www.oxid-esales.com

E-Mail: lars.jankowsky@oxid-esales.com

Fon: +49 761 36889 0